Real-Time Design Patterns for LVC Simulation

Douglas D. Hodson
douglas.hodson@wpafb.af.mil
Simulation & Analysis Facility
ASC/XRA, WPAFB, OH

Timothy E. Menke
timothy.menke@wpafb.af.mil
Simulation & Analysis Facility
ASC/XRA, WPAFB, OH

Rusty O. Baldwin, PhD
rusty.baldwin@afit.edu
Air Force Institute of Technology
WPAFB, OH

ITEA LVC Conference Jan 12-15, 2009

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE **JAN 2009**	2. REPORT TYPE	3. DATES COVERED **00-00-2009 to 00-00-2009**

4. TITLE AND SUBTITLE **Real-Time Design Patterns for LVC Simulation**	5a. CONTRACT NUMBER 5b. GRANT NUMBER 5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER 5e. TASK NUMBER 5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **ASC/XRA,Simulation & Analysis Facility,Wright Patterson AFB,OH,45433**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S) 11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**25**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

Agenda

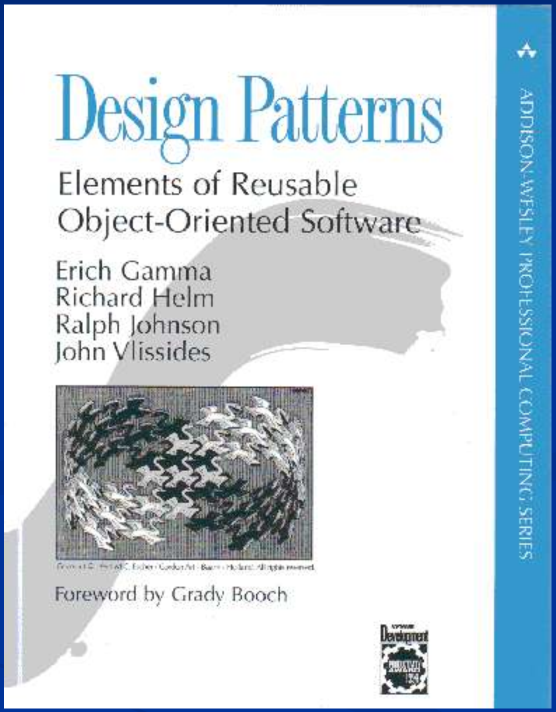
- Frame of Reference
- What is a Design Pattern?
- Real-Time Concepts
- Model-View-Controller
- Components
- Estimating Performance
  - Computing Utilization
  - Rate Monotonic Analysis

Frame of Reference

- Higher Throughput Does NOT Guarantee Real-time Performance
  - Real-time ≠ Fast, Real-time = Predictable
  - HP Computing Oriented Towards Improving Throughput
    - Completing More Work/Unit Time
  - Real-time Computing Oriented Towards Improving Timeliness
    - Completing Work on Time
    - Deterministic Response Times
- LVC Applications
  - Involves Real People and Systems – Hardware-in-the-loop
  - A Class of Interactive Applications
  - Falls into the Domain of Real-time Computing
- Paper Oriented at Incorporating Real-time Concepts into Well Documented Design Patterns
  - Introduces Concepts that Leverage Multiple CPUs to Improve Real-time Performance

What is a Design Pattern?

- Is a General Reusable Solution to a Commonly Occurring Problem in Software Design.
- It is Not a "Finished" Design that can be Transformed Directly into Code.
- It is a Description or Template for How to Solve a Problem that can be Used in Many Different Situations.
- Gained Popularity After Gamma's Book was Published in 1994.
- How Do we Apply this to Modeling and Simulation Execution?
  - What About Real-time Issues?



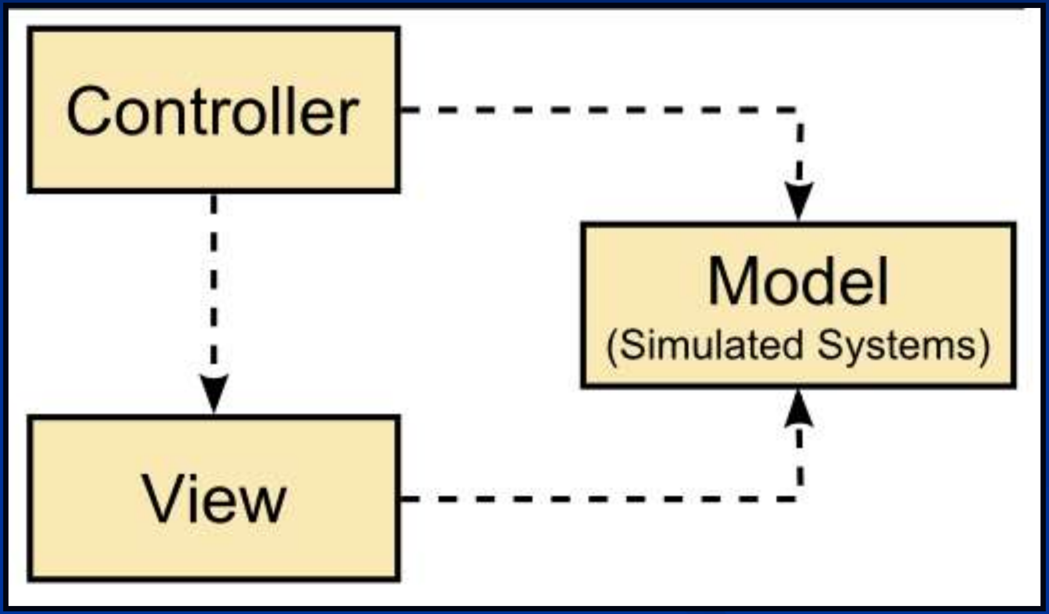
Real-Time Concepts

- Software Systems with Timing Constraints
  - Virtual Simulation
    - Executes in Sync with Wall-clock
  - Interaction Response Characteristics
    - Time to Generate Outputs from Inputs
- Real-time Paradigm: Partitioning of Code
  - A Design Pattern of Sorts!
  - Foreground
    - Jobs that have Time Deadlines
      - Example: Model Mathematics, Redrawing Interface Displays, etc
    - Executed on a Periodic Basis
      - Example: 50 Hz for Models, 20 Hz for Interface Displays
  - Background
    - Jobs Without Timing Constraints
      - Example: Logging Data to a Hard Drive
    - Execute Whenever Possible. (But Must Get Done at Some Point)

MVC Pattern

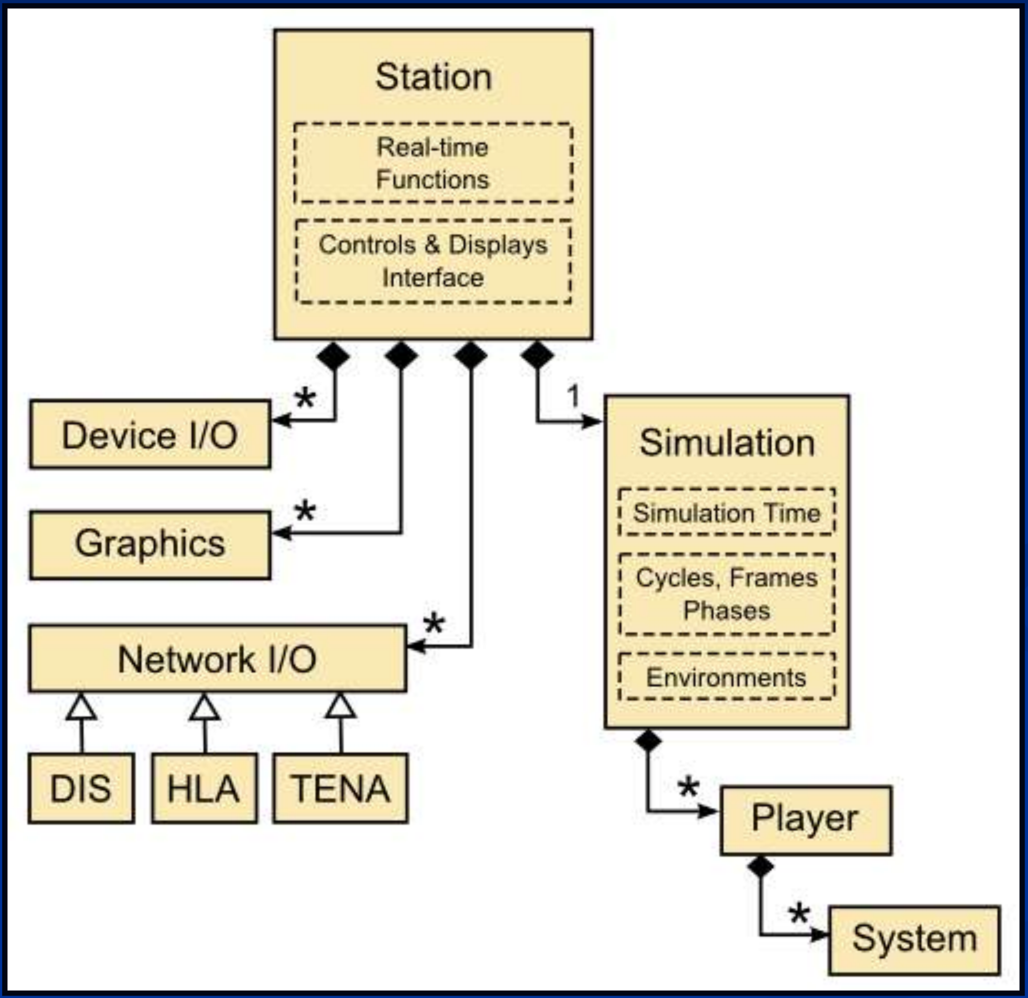
(Tailored for LVC)

MVC Pattern



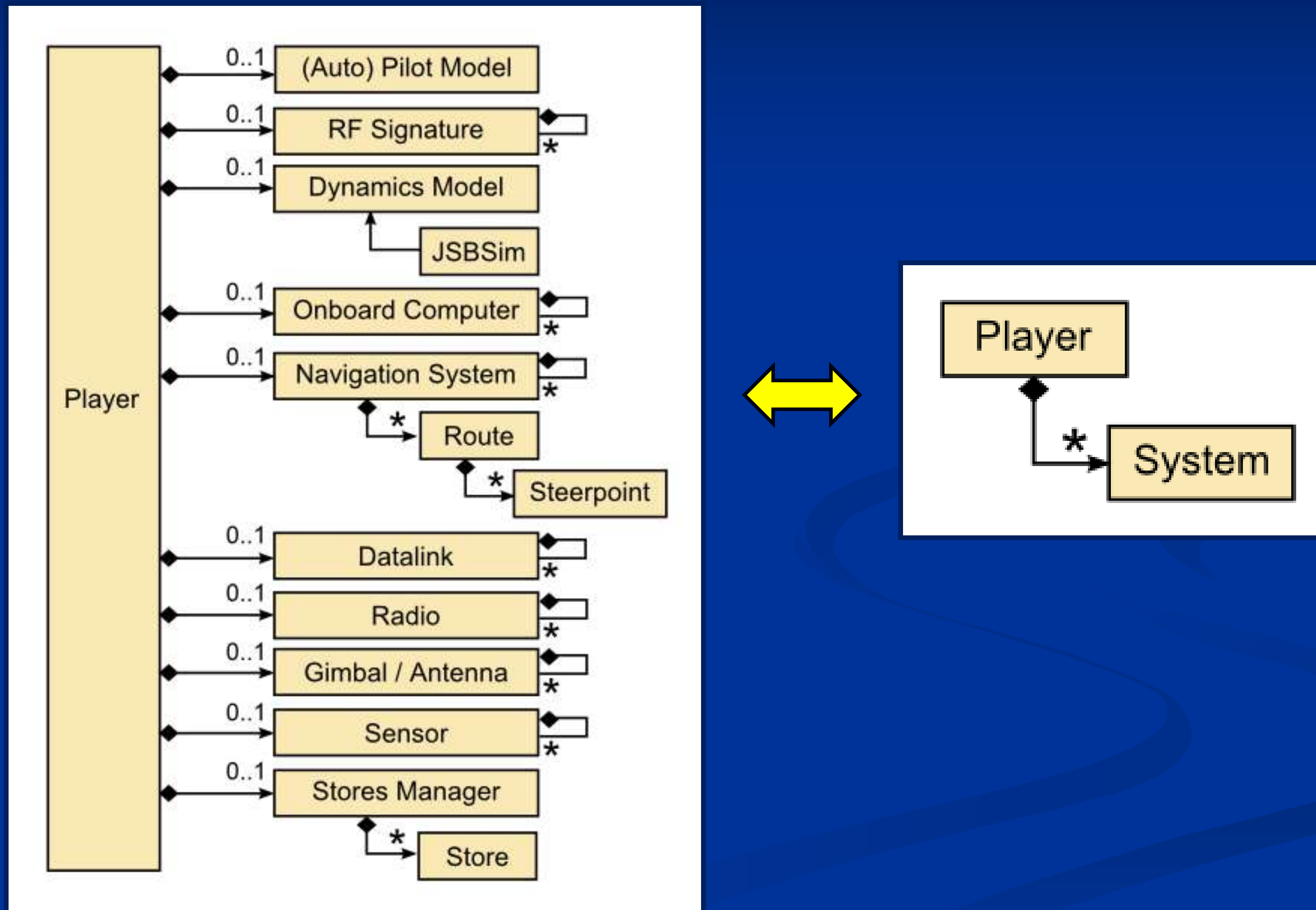
- Model is the Application's Domain Logic
  - It's the Simulation!
- View is the Application's Graphical Displays
- Controller Connects Model to View(s)

Adapted MVC Pattern



- Asynchronous Execution of Simulated System, Graphics and Network I/O
- Architecture Maps to Real-time Design Paradigms
  - Good “Fit” for LVC Requirements
- Leverages Multi-CPU & Multi-core Systems

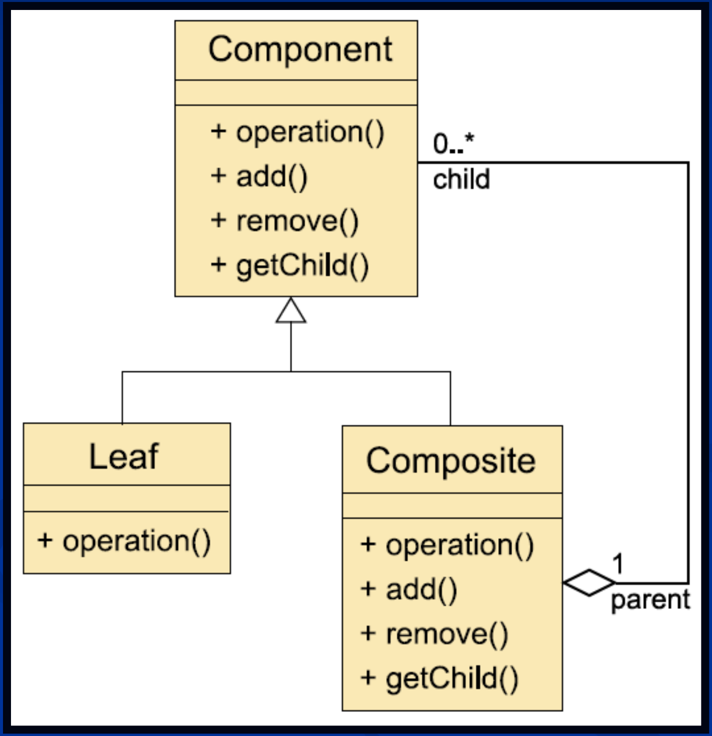
Player Pattern



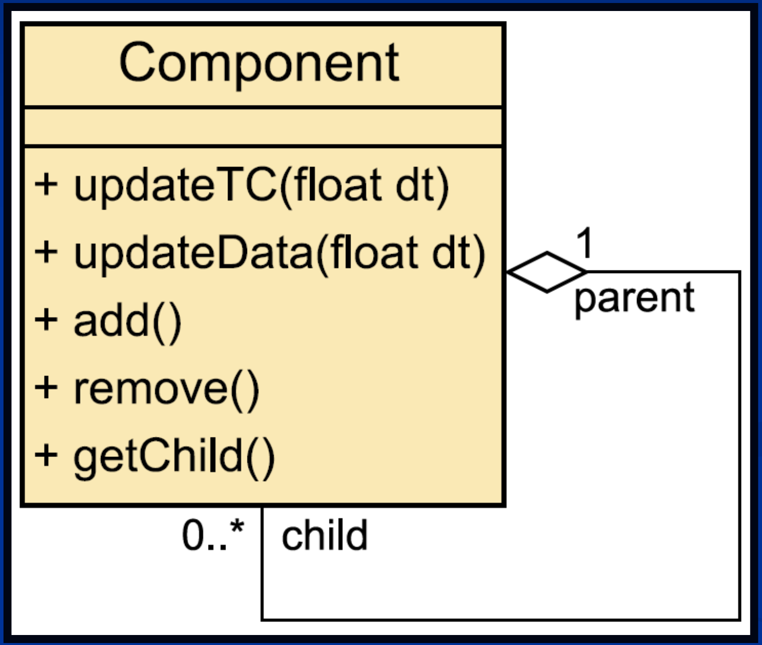
Component Pattern
(Tailored for LVC)

Composite Pattern

- Pattern from Gamma
  - 3 Classes
  - It has "Problems"
- We Modify
  - Models are <u>Always</u> Abstractions
  - Delete "Leaf"
  - operation() Method Replaced by Foreground & Background Methods
  - <u>A Component is a Composite</u>

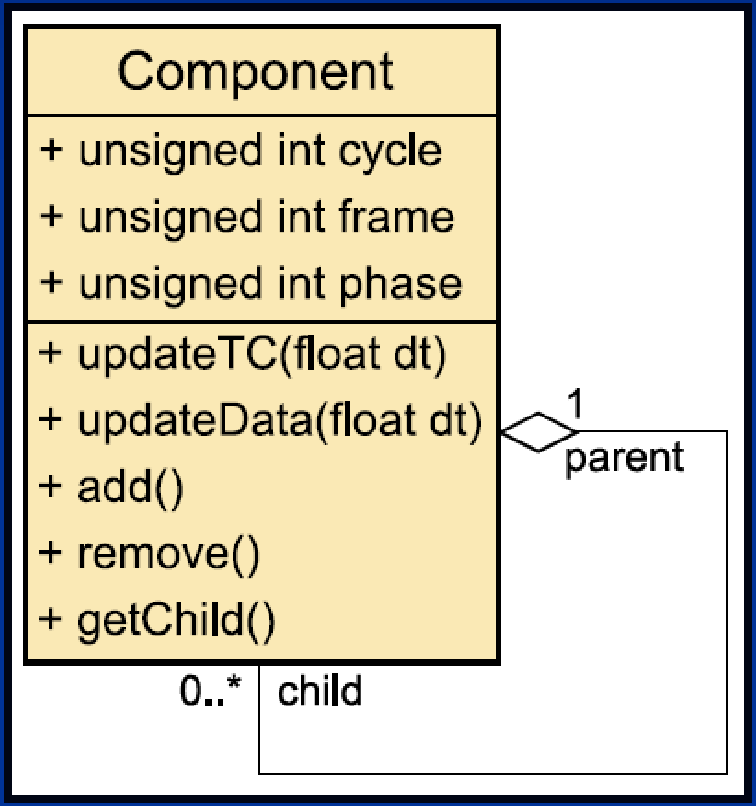


Real-Time Component For Hierarchical Modeling



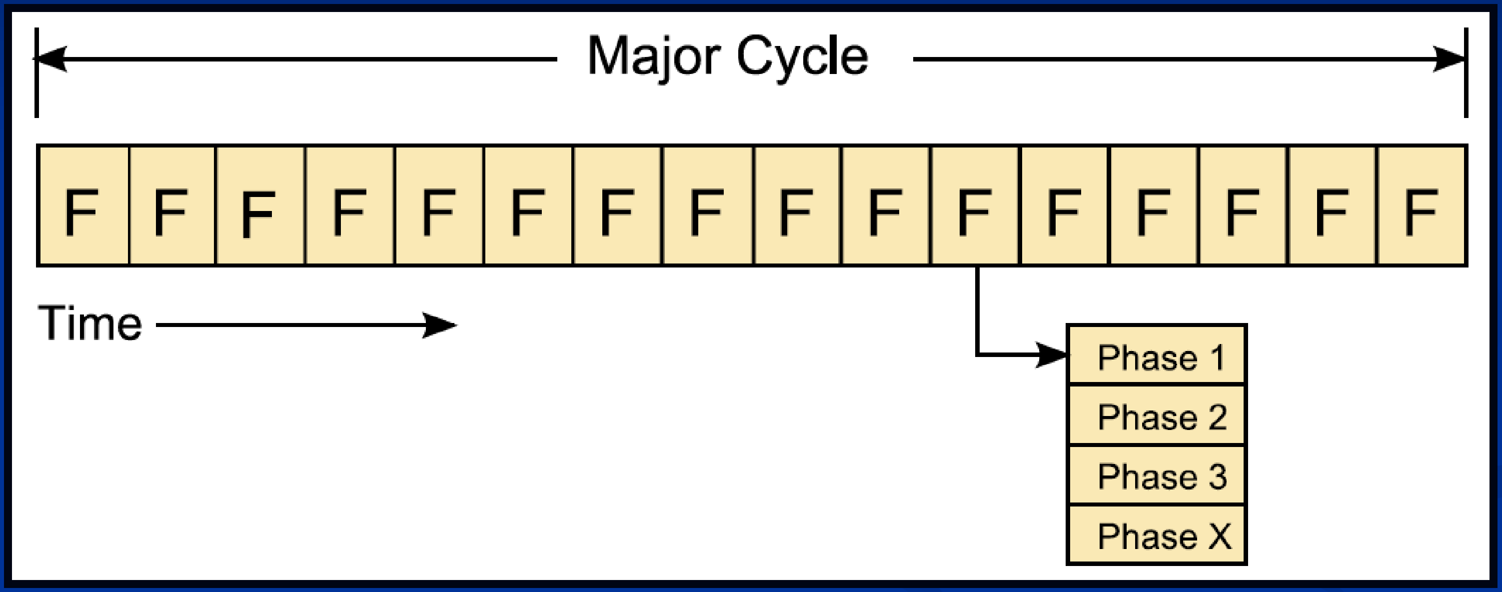
- updateTC – Update Time Critical Jobs
- updateData – Background Processing

Support for Cyclic Scheduler



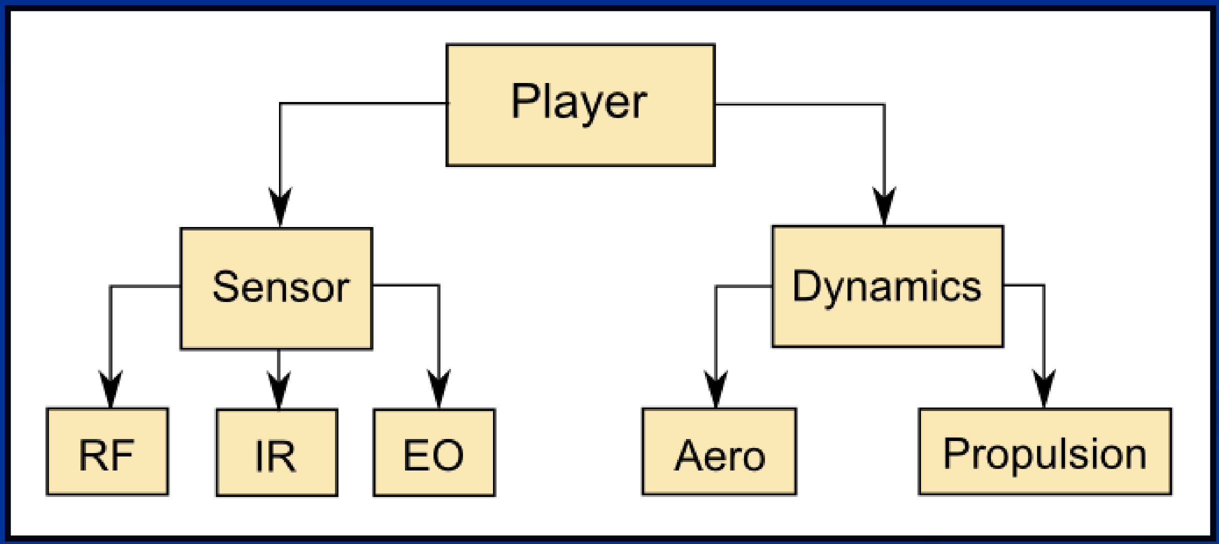
Added Attributes to Support a Cyclic Scheduler

Scheduling Model Code
(Cyclic Scheduler)

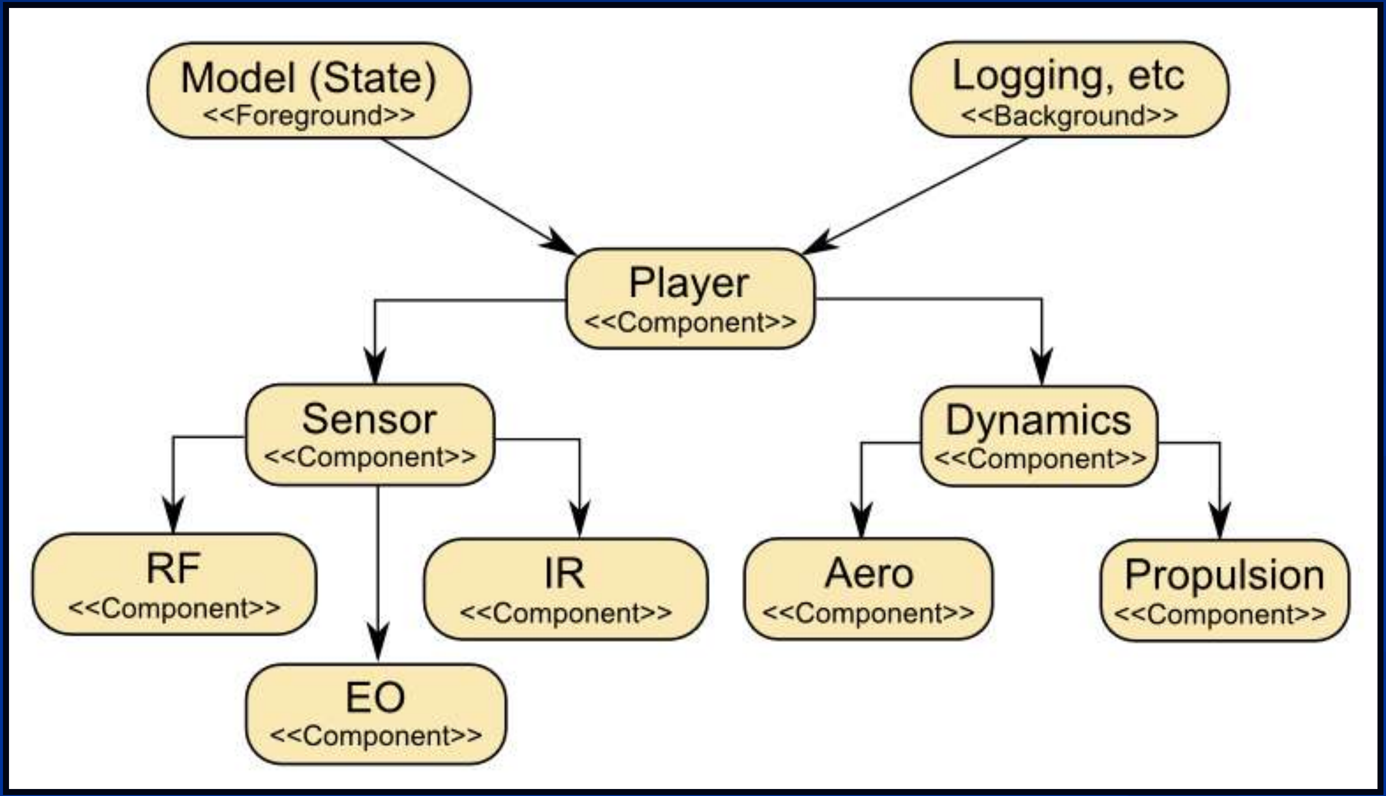


- Provides More Modeling Flexibility
  - Code can be Scheduled to Execute in Different Frames
  - Phases Provide Order
    - Example: Player Dynamics Computed in First Phase of Each Frame
    - Example: Sensor Calculation Performed in Second Phase

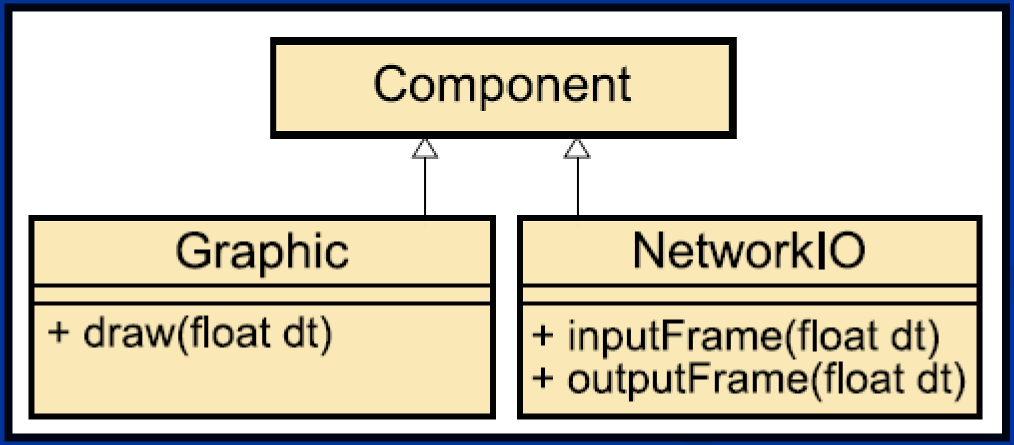
Hierarchical Model of a Player



Player Implementation
(An Object Tree)



Extending Component
(Graphics and I/O)



- Specialized Classes for Graphics and Interoperability
  - Each Organized to Support An Thread

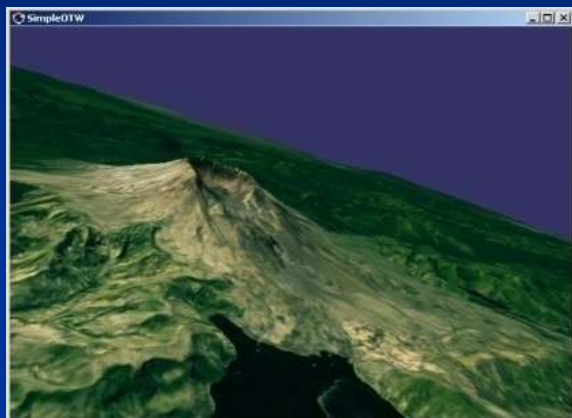
A Simulation

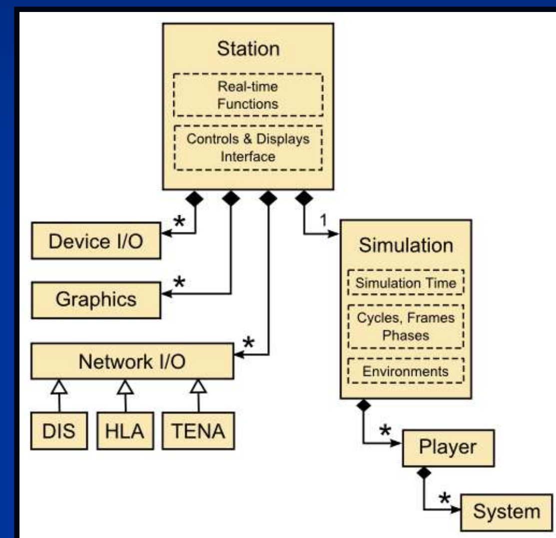
- Simulation Application
  - Simulation Consists of a List of Players (Object Trees)
  - Interfaces (Graphics/IO) Implemented with Specialized Components (Object Trees)
- Each Tree can be Associated with a Thread
- For Example
  - Time Critical Simulation State Space Updates can be Associated with a High Priority Thread
  - Graphics and Network “View-Controllers” can be Associated with Individual Threads as Needed
  - Background Processing (Logging Data to Disk, etc) is Executed when No Other Higher Priority Thread is Ready

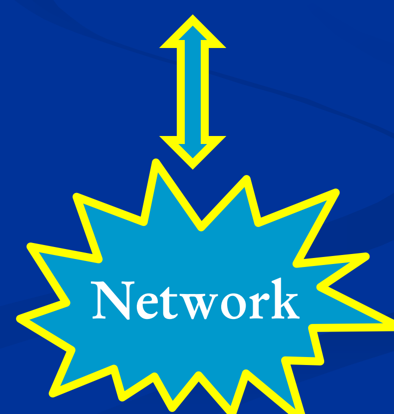
Beyond 4 CPUs

- Typically, the Simulation Serially Processes the Player List
- Right Level of Granularity for Additional Parallelism
- Independent Threads can Process Individual Players
  - For Example: 2 CPUs can Each Process Half the Player List
- But! Care Must be Taken to Sync with Phases
  - Data Dependencies Limit Scalability
- Must Determine Breakpoint where Performance of Additional Parallelism Exceeds Cost of Overhead to Implement

Putting it All Together







Estimating Performance

Total Utilization

$$\boxed{U = \sum_{i=1}^{n} {}^{e_i}/_{p_i}}$$

Rate Monotonic Analysis

Theorem 1 (Rate Monotonic) [8] *Given a set of periodic tasks and preemptive priority scheduling, then assigning priorities such that the tasks with shorter periods have higher priorities (rate-monotonic), yields an optimal scheduling algorithm.*

Theorem 2 (RMA Bound) *Any set of n periodic task is RM schedulable if the processor utilization, U, is no greater than* $n(2^{1/n} - 1)$.

Example Calculation

$$U = {}^{e_{sim}}\!/_{p_{sim}} + {}^{e_{draw}}\!/_{p_{draw}} + {}^{e_{net}}\!/_{p_{net}}$$

If this computed utilization is not greater than the RMA bound, $n(2^{1/n} - 1)$, or 780 ms/s (for $n = 3$), then the system is schedulable.

Summary

- LVC Simulations are Real-time Systems
- Design Patterns Provide a General Solution to a Commonly Occurring Problem
- Adapted Both the MVC and Component to the Domain of LVC
  - By Incorporating Real-time Concepts
- Big Picture
  - Organized Software Code So Performance Estimates Can be Made
- EAAGLES Framework is Based Upon these Patterns